Seguimiento de dispositivos Android

Programación de Sistemas

Curso 2014-2015

Segundo Cuatrimestre

7 de abril del 2015

Canosa Oroña, Juan Manuel - jmtatic@gmail.com

Trabazo Sardón, Diego - diego.trabazo@udc.es (contacto)

Índice

Motivación y objetivos

El proyecto a desarrollar tiene como objetivo crear una aplicación que permita el seguimiento de dispositivos móviles, basados en tecnología Android, por el territorio. La motivación surge de la necesidad, patente en múltiples situaciones, de disponer de la información veraz y una herramienta que permita consultarla y gestionarla, con el fin de localizar personas, animales u objetos y surge tanto en el ámbito personal como profesional. Se parte con la idea de diseñar y construir los módulos básicos que permitan al dispositivo enviar la información generada a un servidor central. Éste la recoge y pone a disposición de aquellas instancias de la aplicación que actúen como clientes. En los clientes se pueden implementar varias funcionalidades para llevar a cabo la explotación de la información, siendo de las más evidentes la visualización en un plano de las últimas posiciones o recorridos realizados. Se trata con esta propuesta de aportar una solución a esta problemática y acercarse a un mercado que puede no conocer este tipo de tecnologías.

Existen ya algunas alternativas en este ámbito que se pueden encontrar, por ejemplo, mediante una búsqueda en el *Play Store* de Google. La empresa Life360[1] comercializa un servicio que permite seguir la pista a familiares y otros grupos de interés. Slash Idea[2] tiene un servicio para llevar el control de una flota de vehículos. Esocial[3] vende una aplicación, *Triplog*, que permite gestionar una flota de vehículos tanto a nivel personal como profesional. Son todas ellas empresas estadounidenses con un producto interesante pero sin soporte en castellano para sus clientes.

1 https://www.life360.com/
2 http://www.mycartracks.com/
3 https://triplogmileage.com/

Análisis preliminar de requisitos

La aplicación constará de dos modos de funcionamiento. Por una parte una operativa como generador de datos geográficos y otra como utilizador y/o visualizador de dichos datos. Estos dos contextos, permiten toda una gama de uso y distintos enfoques. Por supuesto la *app* funcionará sobre el sistema Android, con un nivel de API todavía por establecer. Se presupone que existirá en el dispositivo alguna conexión a Internet como canal de transmisión y recepción de los datos.

El flujo de información generado no será volcado directamente a la instancia o instancias que actúen como visualizadoras o gestoras en un momento dado. Es necesario contar con una infraestructura intermedia que reciba los datos, y controle quién puede acceder y cómo. Para resolver este problema de intercambio de datos se propone el uso de librerías Open Source y tecnologías estándar sobradamente conocidas, que se detallan a continuación.

El módulo servidor utilizará la librería ROME[4] para generar entradas RSS[5] que se almacenarán en un servidor web. A su vez los clientes, previa solicitud inicial, podrán suscribirse a los feed deseados, representando cada uno a un rastreado. Se utilizará también dicha librería para realizar la lectura y procesado de la información en el módulo cliente. Para la comunicación vía protocolo HTTP se propone la librería HttpURLConnection[6].

El servidor web propiamente dicho podrá utilizar cualquier implementación libre, por ejemplo Apache. Para permitir la recepción de los feeds y su publicación se propone el uso de Python como lenguaje de *scripting*. Nótese que en el servidor se podría gestionar el contenido sindicado con la granularidad deseada, en términos de seguridad de acceso a la información, dependiendo de las necesidades y el tiempo disponible. Además no hay, en principio, límites a la cantidad de información que será almacenada y que, por consiguiente, podrá ser recuperada por un cliente interesado.

Otro aspecto a considerar es la creación de las listas de seguimiento y cómo gestionarlas. Dado que cada instancia tendrá asociada la dirección URL de su feed, la acción de seguir a otro dispositivo se realizará mediante la lectura de su feed. De manera que una lista de seguimientos no será más que una colección de URLs a las cuales se tendrá acceso.

4 https://github.com/rometools/
5 http://en.wikipedia.org/wiki/RSS
6 http://developer.android.com/reference/java/net/HttpURLConnection.html

Funcionalidades

1. Activación/desactivación de la generación de datos geográficos en el dispositivo local.
2. Listas de seguimiento para incluir o excluir aquellos dispositivos que resulten de interés.
3. Visualización de las posiciones de los rastreados en el tiempo, siendo este el momento actual o pretérito, según el acceso a la colección de datos.
4. Alertas de proximidad, que abarcará:
    1. Proximidad geográfica entre instancias de la aplicación.
    2. Proximidad de una instancia a un lugar definido previamente.
5. Alertas de velocidad máxima, mínima, media, etc. en un intervalo de tiempo.
6. Gestión de seguimientos pudiendo:
    1. Generar códigos de seguimiento para compartir con otros dispositivos.
    2. Solicitar seguimientos a otros dispositivos.
    3. Aceptar/denegar peticiones de seguimiento.

Prioridades, dependencias y orden

En primer lugar ha de implementarse la parte servidora. Configurar Apache y escribir los scripts necesarios para generar el feed RSS haciéndolo disponible en una URL. Lo segundo necesario, ya en la aplicación Android, es el módulo que genera y envía los datos en formato RSS al servidor. A continuación se necesita un módulo que permita gestionar las suscripciones a los feeds de otros cliente y a ser posible una manera de compartir dicha información. Por último, dentro de las características necesarias, la aplicación tendrá que poder leer la información a la que se ha suscrito y explotarla de forma básica con alguna de las funcionalidades propuestas, por ejemplo, poder visualizar sobre un mapa las últimas posiciones conocidas de los clientes seguidos. Las demás funcionalidades son secundarias, no obstante, tendrían prioridad aquellas relacionadas con la explotación de los datos, lo que el cliente puede utilizar de la aplicación.

Responsabilidades

En este momento inicial del desarrollo el trabajo es realizado en sesiones conjuntas. Más adelante se buscará definir bloques de tareas que se puedan contribuir de forma independiente y que, por consiguiente, sean más fácilmente asignables a cualquiera de los integrantes del grupo de trabajo. Para la coordinación se dispone de herramientas como Google Docs, mensajería instantánea y para el desarrollo se utiliza el Sistema de Control de Versiones Subversion.

Iteraciones, hitos y entregables

El proyecto se desarrollará a lo largo del cuatrimestre en cuatro fases:

1. Primera iteración. No implicará desarrollo de software sinó la elaboración inicial del presente documento, explicando en qué consiste el proyecto en sí y detallando su desarrollo futuro.
2. Segunda iteración. Se alcanzará el primer hito: la aplicación publicará correctamente contenido en una URL en el servidor, en formato RSS. Para la primera aplicación se pretenderá que se pueda probar con un servidor en una máquina virtual y un widget sencillo en el terminal móvil para demostrar que la comunicación está resuelta. Se podrá probar por tanto dicha comunicación.
3. Tercera iteración. Los datos GPS obtenidos. La aplicación publica en su feed (y cada instancia en un feed correspondiente discriminándose utilizando el IMEI o un mecanismo similar).
4. Entrega final. Explotación atractiva de los datos, por ejemplo, visualización de los últimos datos disponibles sobre un mapa.

Incidencias y planes de contingencia.

A continuación se muestran las incidencias que se han ido encontrando durante el desarrollo de la aplicación.

Antes del comienzo

Sin haber probado nunca las librerías que se desean utilizar, el punto débil de utilizar un sistema basado en RSS y un servidor web es que no se conoce como resultará en la práctica o el trabajo inicial necesario.

Hasta el primer hito

ROME no funciona en Android. Tan pronto como se quiso probar los primeros ejemplos de la documentación de ROME se vio que la librería no funcionaba[7]. ROME utiliza las clases *java.beans* del API estándar de Java pero que no están disponibles en el API de Android. Por suerte la solución vino de la mano del proyecto Apache Harmony que reempaqueta dicha librería. Concretamente se optó por otra[8] versión que sustituye el espacio de nombre habitual por otro diferente. Hacer esto obligó a sustituir en ROME todos los imports de *java.beans* a *com.googlecode.openbeans*. Tras recompilar correctamente ROME e integrar el nuevo paquete al proyecto Android aparecieron más dependencias, JDOM y especialmente SLF4J que tampoco está disponible en Android pero se puede obtener desde la web del propio proyecto[9]. Finalmente, con estas adiciones ROME sí funciona.

7 http://therealdanvega.com/blog/2010/05/23/parsing-rss-feeds-in-android
8 https://code.google.com/p/openbeans/
9 http://www.slf4j.org/android/

Hasta el segundo hito

El objetivo de que cada dispositivo publicase en su propio feed funciona en este momento de la forma ideada. Ha sido necesario modificar ligeramente el script utilizado en el servidor de tal manera que una vez obtenido el cuerpo de la query, extrajese de él el IMEI para poder volcar los datos a su directorio definitivo. Sobre la decisión inicial de utilizar el IMEI como discriminante de los dispositivo, parece que no resulta ser una buena decisión. Un motivo es que para obtenelo es necesario que la aplicación adquiera el permiso READ_PHONE_STATE que da acceso a demasiada información o, dicho de otra manera, resulta muy intrusivo para el usuario. Por tanto sería interesante buscar alguna manera de reemplazarlo, por ejemplo con el Google+ Login API[10].

Un detalle ha considerar, y que ha surgido durante el desarrollo, es cómo determinar si es la primera vez que un dispositivo publica información o no. Para resolverlo se ha decidido, como primera aproximación, hacer una consulta sobre la URL de destino del feed en el servidor. Si hubiese datos, se leerían y procesarían directamente en el dispositivo, añadiendo a esos datos existentes la nueva entrada. La ventaja de esta aproximación es que se libera al servidor de trabajo y los dispositivos pasan a ser más independientes. Por otra parte si el cliente está generando datos se pueden ir almacenando localmente y actualizar por la red a una velocidad distinta, transmitiendo de esta manera menos información duplicada. La generación de los datos y su publicación quedan desligados. Sin embargo la implementación correspondiente no está todavía realizada. Dado que es básica para la funcionalidad de la aplicación, debe ser realizada en primer lugar.

Otro de los objetivos para este segundo hito era la obtención de los datos de GPS. Este objetivo no se ha alcanzado, por tanto, y siendo básico para la funcionalidad de la aplicación, es necesario resolverla cuando antes.

Diseño: arquitectura

El proyecto se basa en la arquitectura cliente/servidor. Durante el desarrollo el servidor se basa en una máquina virtual con sistema operativo Debian con el software necesario instalado.

Se emplearán actividades y fragmentos repartiéndose la carga de visualización y gestión de los datos.

10 http://stackoverflow.com/questions/1972381/how-to-programmatically-get-the-devices-imei-esn-in-android